

# On insertion-deletion systems over relational words

Igor Potapov

*potapov@liverpool.ac.uk*

Department of Computer Science, University of Liverpool  
Ashton Building, Ashton Street, Liverpool L69 3BX, UK

Olena Prianychnykova

*olena.prian@tu-ilmenau.de*

Fachgebiet Automaten und Logik, Instituts für Theoretische Informatik,  
Technische Universität Ilmenau, D-98684, Ilmenau, Germany

Sergey Verlan

*verlan@u-pec.fr*

LACL, Departement Informatique, Université Paris Est Créteil  
61, av. General de Gaulle, 94010 Créteil, France

October 13, 2015

## Abstract

We introduce a new notion of a relational word as a finite totally ordered set of positions endowed with three binary relations that describe which positions are labeled by equal data, by unequal data and those having an undefined relation between their labels. We define the operations of insertion and deletion on relational words generalizing corresponding operations on strings. We prove that the transitive and reflexive closure of these operations has a decidable membership problem for the case of short insertion-deletion rules (of size two/three and three/two). At the same time, we show that in the general case such systems can produce a coding of any recursively enumerable language leading to undecidability of reachability questions.

**AMS Subject Classification:** F.4.2 Grammars and Other Rewriting Systems, F.4.3 Formal Languages

**Keywords:** Infinite alphabet, relational words, insertion-deletion, membership

## 1 Introduction

Nowadays there is a sufficiently broad research activity in the area of logic and automata for words and trees over infinite alphabets. It is mainly motivated by the need to analyse and verify infinite-state systems, which for example can use

infinite alphabet of natural numbers 1, 2, 3, . . . instead of finite number of symbols like a, b, c. In the seminal paper of M. Kaminski and N. Francez [6] a very restricted memory structure of the automaton (Register Automaton) working with words over infinite alphabets was introduced. The register automaton is operating by keeping a finite number of symbols (from the working tape) in its memory and making their comparison to other observed symbols. The model allows recognising a large class of languages over infinite alphabet and at the same time is not taking any advantage of its memory capabilities beyond what is needed for that purposes. Later, more models including automata on data words, data trees, pebble automata, etc have been considered in the realm of semistructured data, timed automata and extended temporal logics [10, 15]. Following motivation from program verification, analysis of XML query languages and other systems operating explicitly with data values, the research was focused on characterization of automata models and logics manipulating data in terms of expressive power and decidability.

Comparing to language recognizers, more complex systems operating with words over infinite alphabet may require updating them in addition to the operations of comparison between symbols. Obviously, unrestricted and very general rules allowing rewriting over arbitrary infinite alphabet are too powerful making most of the computational problems to be undecidable [3]. On the other hand there are existing fragments of rewriting systems over infinite alphabet with a decidable word problem (i. e. the algorithmic problem of deciding whether two given representatives represent the same element of the set). These examples are not limited to classical computer science objects, but also include examples from other areas. One of such examples is unknotedness and equivalence of knots, where words over infinite alphabet are Gauss words (or Gauss diagrams) and the system of rewriting rules is a set of Reidemeister moves represented by insertion/deletion and swapping some of the symbols on Gauss words [14]. While the set of the Reidemeister moves is quite powerful the word problem for such rewriting rules on Gauss words is decidable following algorithms from combinatorial topology.

In this paper we aim to extend the concept of the computations on words over infinite alphabet but preserving original idea of indirect references, i.e. computations where we only make comparison between positions in our data without explicit references to their values. In particular we extend the notion of a word on an infinite alphabet by allowing the equivalence relation to be defined on a subset of the set of positions of a word. A new notion of *relational word* is defined as a finite set of positions equipped with binary relations that describe which positions are labeled by equal and non equal data, while for some pairs of positions the relation between their labels could be undefined. Similar idea of representing data over a finite alphabet as a set of relations was also named as a “relational code” can be found in [5], which generalize “partial words” in the area of nonstandard stringology [11] and DNA sequence processing [7]. Another example can be found in [1] where authors introduce a first-order logic  $FO(\sim, <, +1)$ , where for every formula  $\varphi$  in this logic the set  $L(\varphi)$  is the set of data words that satisfy a sentence  $\varphi$ . This approach also allows the specification using a kind of a template, the main difference from the model proposed in the present paper being the unbounded length of the specified string (in the case of the model from this paper all the words corresponding to the same relational word have the same fixed length). All of the above models study words and

languages described by some relations, but do not define any rewriting on these structures.

In the concept of *relational word*, instead of a particular words over an infinite alphabet, we can operate with templates that may represent finite or infinite languages depending on a choice of the alphabet. This gives us an opportunity to define rewriting of data on a new conceptual level focusing only on the operations of rewriting based on existing relations in data and abstracting from actual data on which we may operate.

The rewriting system on data is interesting both from theoretical and practical aspects, see [2, 3, 4]. In this paper we consider a very natural rewriting system, motivated by the knot theory, in which only insertion and deletion operations are defined. Also insertion and deletion are considered to be the basic operations in DNA processing and RNA editing [13] and in the context of an infinite alphabet insertion-deletion systems are important for reasoning about recursive sequential programs, multithreaded programs, parametrized and dynamic networks of processes, etc [3]. In particular we study the membership problem: for a given set of insertion/deletion operations defined by relational words decide whether a relational word  $w$  can be derived from an empty word. We show that for any system which inserts 2 symbols and deletes 3 symbols or vice versa the membership is decidable. We also show that this does not hold anymore for longer insertion and deletion rules – in this case the membership and the word problems are undecidable.

## 2 Preliminaries

### 2.1 Relational words

A finite sequence of elements of a finite alphabet  $\Sigma$  is called a finite word over  $\Sigma$ , or just a word. We denote by  $\Sigma^*$  the set of words over  $\Sigma$  and by  $\Sigma^+$  the set of nonempty words. The empty word is denoted by  $\varepsilon$ .

Let  $\Delta$  be an infinite set. A word over an infinite alphabet  $\Delta$  is a finite sequence of elements of  $\Delta$  [6, 8, 10, 15]. Elements of a finite alphabet  $\Sigma$  are defined explicitly and could be accessed directly, while elements of an infinite alphabet  $\Delta$  could be only tested for equality. Then a word over an infinite alphabet may be viewed as a finite totally ordered set of positions endowed with an equivalence relation.

A well-known example of words over an infinite alphabet are data words[8, 15]. Let  $\Sigma$  be a finite alphabet and  $D$  be an infinite set of data values. A data word is a finite sequence over  $\Sigma \times D$ , i.e., in a data word each position carries a label from a finite alphabet and a data value from some infinite domain. A data word may be viewed as a word over finite alphabet  $\Sigma$  with an equivalence relation on the set of its positions [1].

Now the idea of this paper is to extend the notion of a word over an infinite alphabet by allowing the equivalence relation to be defined on a subset of the set of positions of the word. We define a relational word as a finite set of positions equipped with binary relations that describe which positions are labeled by equal and by unequal data, while for some pairs of positions the relation between their labels is not defined.

A relational word can be viewed as a kind of a template. For an alphabet  $A$

(finite or infinite) a relational word  $W$  defines a language  $L_A(W) \in A^*$  which is the set of all words  $w = a_1a_2\dots a_n$ , where  $a_i \in A$ ,  $1 \leq i \leq n$ , with  $n$  being the length of  $W$ , such that for every pair of positions  $i$  and  $j$  in  $W$  we have

- if  $(i, j)$  belongs to the equality relation, then  $a_i = a_j$
- if  $(i, j)$  belongs to the inequality relation, then  $a_i \neq a_j$

We remark that if any pair of positions of a relational word  $W$  is a member of a relation (equality or inequality), then  $L_A(W)$  can be identified with any element  $w \in L_A(W)$ , as based on  $w$  and  $A$  it is possible to reconstruct  $L_A(W)$ . This gives the possibility to represent a relational word using ordinary words, where same letters represent the equality relation and different letters the inequality one. If it will be clear from the sequel we will not indicate the index  $A$  in  $L_A(W)$ .

With every relational word  $W$  we can associate a graph  $G^W = (Q, T)$  and an edge labeling function  $Lab_{G^W} : T \rightarrow \{0, 1\}$  such that

- $Q = \{q_1, q_2, \dots, q_n\}$  is an ordered set of nodes,  $n$  is the length of  $W$ ,
- $T \subseteq Q \times Q$  is the set of edges such that  $(q_i, q_j) \in T$  iff there is a relation (equality or inequality) between positions  $i$  and  $j$ .
- $Lab_{G^W}$  is defined as follows
  - $Lab_{G^W}(q_i, q_j) = 1$  if the labels of the positions  $i$  and  $j$  are equal,
  - $Lab_{G^W}(q_i, q_j) = 0$  iff the labels of the positions  $i$  and  $j$  are not equal.

We will use the following convention for the graphical representation of  $G^W$ . The nodes of the graph will be aligned horizontally and the order of nodes taken from left to right will correspond to their ordering within the graph. We will depict edges labeled by 1 below the axis induced by the node alignment and the edges labeled by 0 on the top of it. We also note that for any  $q_i$ , there exist an edge  $(q_i, q_i)$  labeled by 1. In order to simplify the pictures we will not draw corresponding self-loops.

With every relational word  $W$  we associate the matrix  $M^W \in \{0, 1, 2\}^{n \times n}$  where  $n$  is the length of  $W$ , as follows:

$$M^W[i, j] = \begin{cases} 1 & \text{iff the labels of the positions } i \text{ and } j \text{ are equal} \\ 0 & \text{iff the labels of the positions } i \text{ and } j \text{ are not equal} \\ 2 & \text{iff the relation between the labels of the positions } i \text{ and } j \text{ is not defined} \end{cases}$$

**Example 1.** Let us consider the relational word  $W$  of length 4 such that the labels of the first and the third position are equal, the label of the second position is not equal to them, and the relations of the label of the fourth position to all others are undefined. The graph that represents  $W$  and the corresponding matrix are shown on the Fig. 1.

Let  $A = \{a\}$ , then  $L(W) = \emptyset$ .

Let  $A = \{a, b\}$ , then  $L(W) = \{abaa, abab, baba, baba\}$ .

Let  $A = \{a, b, c\}$ , then  $L(W) = \{abaa; abab; abac; baba; babb; babc; acaa; acab; acac; caca; cacb; cacc; bcba; bcb; bcbc; cbca; cbc; cbcc\}$ .

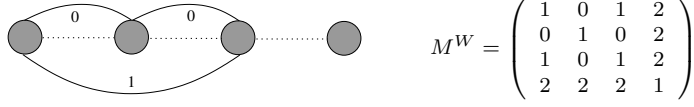


Figure 1: An example of a relational word and the corresponding matrix.

Let  $W$  be a fully defined relational word and  $A$  be an alphabet. Then every word  $w \in L(W)$  is an assignment of equivalence classes to symbols of  $A$ . Hence if the alphabet  $A$  is finite, then the language  $L(W)$  is finite (and thus it is regular). Moreover,  $|L(W)| = C_n^k$  where  $k$  is the number of classes of the equivalence relation and  $n$  is the size of the alphabet  $A$ . If the relational word  $W$  is not fully defined, then for every  $w \in L(W)$  the number  $k$  of different symbols in  $w$  could not be larger than the length of the word  $W$ . Then for every alphabet  $A$  and every relational word  $W$  the set  $L(W)$  is finite if and only if  $A$  is finite.

Now we give a formal definition of a relational word.

**Definition 1.** A relational word is a relational structure  $W = (X^W, E^W, N^W)$  where

- $X^W = (X^W, \prec)$  is a finite totally ordered set;
- $E^W$  and  $N^W$  (for equal and not equal) are binary relations on  $X^W$  such that
  - they are mutually exclusive:  $E^W \cap N^W = \emptyset$ ;
  - $E^W$  is an equivalence relation;
  - $N^W$  is a symmetric relations;
  - for every  $x, y, z \in X^W$ , if  $(x, y) \in E^W$ , then  $(x, z) \in R^W$  if  $(y, z) \in R^W, R \in \{E, N\}$ .

For technical reasons we shall consider the relation  $U^W = X^W \times X^W \setminus (E^W \cup N^W)$  corresponding to an undefined relation between pairs of positions.

We denote by  $|W| = |X^W|$  the length of the relational word  $W$  and by  $W[i]$  the  $i$ -th element from the ordering of  $X^W$ . The empty relational word is denoted by  $\varepsilon$ ,  $|\varepsilon| = 0$ . A relational word  $W$  is fully defined if  $U^W = \emptyset$ .

We denote the set of all relational words by  $\mathbb{RW}$  and the set of all fully defined relational words is denoted by  $\mathbb{FDRW}$ .

**Example 2.** Let us consider the relational word  $W$  from the Example 1. We have that  $X^W = \{x_1, x_2, x_3, x_4\}$ ;  $x_1 \prec x_2 \prec x_3 \prec x_4$ ;  $x_1$  is equal to  $x_3$ ,  $x_2$  is not equal to  $x_1$  and  $x_3$ , the relations between  $x_4$  and  $x_1, x_2, x_3$  are undefined, i.e.,

- $E^W = \{(x_1, x_1), (x_2, x_2), (x_3, x_3), (x_4, x_4), (x_1, x_3), (x_3, x_1)\}$ ;
- $N^W = \{(x_1, x_2), (x_2, x_1), (x_2, x_3), (x_3, x_2)\}$ ;

**Definition 2.** Two relational words  $W$  and  $V$  are equal if  $|W| = |V| = n$ , and for every  $1 \leq i, j \leq n$  we have  $(i, j) \in \begin{cases} E^V & \text{iff } (i, j) \in E^W, \\ N^V & \text{iff } (i, j) \in N^W. \end{cases}$

We also introduce the notion of *contradiction* for relational words. Informally, word  $W$  contradicts word  $V$  if it is impossible to get the same fully defined relational word by instantiating to equality or inequality the relations between undefined positions of  $W$  and  $V$ . If  $W$  contradicts  $V$ , then  $L(W) \cap L(V) = \emptyset$ . Formally we define contradiction as follows.

**Definition 3.** A relational word  $W$  contradicts a relational word  $V$  if  $|W| = |V| = n$ , and there are  $i$  and  $j$  such that  $1 \leq i, j \leq n$  and either  $(i, j) \in E^W$  and  $(i, j) \in N^V$ , or  $(i, j) \in N^W$  and  $(i, j) \in E^V$ .

In order to be able to work not only with a relational word as a whole but with parts of it, we will need a notion of a subword.

**Definition 4.** A relational word  $W$  is a scattered subword of  $V$  if  $X^W \subseteq X^V$  and for every  $x, y \in X^W$  we have  $(x, y) \in \begin{cases} E^W & \text{iff } (x, y) \in E^V, \\ N^W & \text{iff } (x, y) \in N^V. \end{cases}$

A relational word  $W$  is a subword of  $V$  if it is a scattered subword of  $V$  and for every  $x, y \in X^V$  if there are  $y, z \in X^W$  such that  $y \prec x \prec z$ , then  $x \in X^W$ .

**Example 3.** Fig. 2 depicts the above notions.

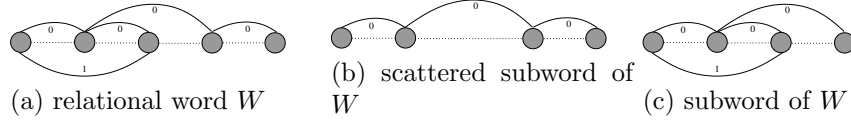


Figure 2: An example of a relational word with its scattered subword and subword

With every relational word  $W$  we associate its numerical characteristics:

1.  $\max FD(W)$  is the length of the longest fully defined scattered subword of  $W$ ,
2.  $\max E(W)$  is the length of the longest scattered subword  $W'$  of  $W$  such that every two elements of that subword are equal, i.e., for every  $x, y \in X^{W'}$  we have  $(x, y) \in E^W$ ,

**Example 4.** Consider  $W$  from the Example 3 we have  $\max FD(W) = 3$ ,  $\max E(W) = 2$ ,  $\max N(W) = 2$ .

## 2.2 Insertion-deletion systems on relational words

**Definition 5.** An insertion-deletion scheme  $S$  is a pair  $S = (INS, DEL)$  where  $INS \subseteq \mathbb{FDRW}$  is the set of insertion rules and  $DEL \subseteq \mathbb{FDRW}$  is the set of deletion rules [9, 13].

The insertion-deletion scheme  $S = (INS, DEL)$  is called *simple* if it contains only one insertion rule and only one deletion rule, i.e.,  $INS = \{I\}$ ,  $DEL = \{D\}$  where  $I, D \in \mathbb{FDRW}$ .

We denote by  $I_n D_m$  the set of all simple insertion-deletion schemes such that the length of the insertion rule is  $n$  and the length of the deletion rule is  $m$ .

Now we define the operations of insertion and deletion operations on relational words.

Informally, given  $W, V \in \mathbb{RW}$  we understand the single-step insertion relation  $W \xrightarrow[S]{\text{ins}} V$  as follows (Fig. 3): to obtain  $V$ , we take  $W$  and  $Y \in \text{INS}$  and “insert”  $Y$  as a subword between any two symbols of  $W$ . We assume that for every pair  $(x, y)$ , where  $x$  is a symbol of  $W$  and  $y$  is a symbol of  $Y$ , the relation between  $x$  and  $y$  is undefined.

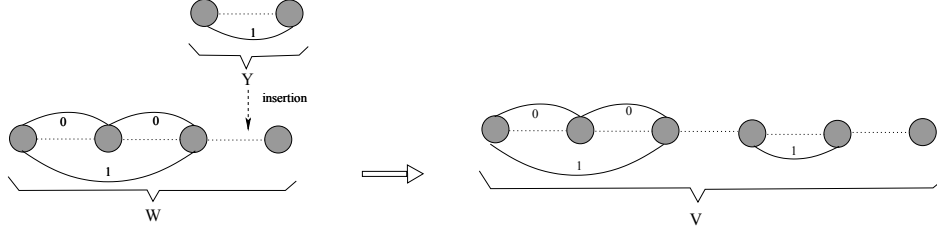


Figure 3: The single-step insertion relation  $W \xrightarrow[S]{\text{ins}} V$

In matrix notation the operation from Fig. 3 can be represented as:

$$\begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 2 \\ 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 1 \end{pmatrix} \xrightarrow[Y]{\text{ins}_3} \begin{pmatrix} 1 & 0 & 1 & 2 & 2 & 2 \\ 0 & 1 & 0 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 2 & 2 \\ 2 & 2 & 2 & 1 & 1 & 2 \\ 2 & 2 & 2 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 & 1 \end{pmatrix}, \text{ where } Y = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Formally, we define this relation as follows.

Consider the function  $s_{k,m} : \mathbb{N} \rightarrow \mathbb{N}$ ,  $k, m \in \mathbb{N}$  defined as follows

$$s_{k,m}(i) = \begin{cases} i & \text{if } 1 \leq i \leq k, \\ i + m & \text{otherwise} \end{cases}$$

**Definition 6.** The single-step insertion relation on  $\mathbb{RW}$  that is induced by  $S = (\text{INS}, \text{DEL})$  is defined as follows. For any  $W, V \in \mathbb{RW}$ ,  $Y \in \text{INS}$  and an integer  $0 \leq k \leq |W|$  we have  $W \xrightarrow[Y]{\text{ins}_k} V$  iff

- $(i, j) \in R^W$  implies  $(s_{k,m}(i), s_{k,m}(j)) \in R^V$ , where  $m = |Y|$  and  $R \in \{E, N\}$ ,
- $(i, j) \in R^Y$  implies  $(i + k, j + k) \in R^V$ , where  $1 < i, j \leq |Y|$ ,  $R \in \{E, N\}$ .

If we are not interested by the site of the insertion or by the concrete insertion rule then we will write  $W \xrightarrow[S]{\text{ins}} V$ , meaning that there exists  $Y \in \text{INS}$  and  $k \geq 0$  such that  $W \xrightarrow[Y]{\text{ins}_k} V$ .

**Definition 7.** The insertion relation on  $\mathbb{RW}$  that is induced by  $S = (\text{INS}, \text{DEL})$  is the reflexive, transitive closure of  $\xrightarrow[S]{\text{ins}}$  and is denoted by  $\xrightarrow[S]{\text{ins}^*}$ .

Now we explain the deletion relation. Informally, the application of the deletion rule  $W \xrightarrow[S]{\text{del}} V$  consists of two steps: expansion and deletion (Fig. 4).

First, we have to find a subword  $Y'$  in the relational word  $W$  that does not contradict to a relational word  $Y \in DEL$  and to “expand” it to  $Y$ : for every symbols  $x$  and  $y$  in  $Y'$  such that the relation between them is undefined, we set this relation to be the same as the relation between the corresponding symbols of  $Y$  (a thick line on Fig. 4). In order to preserve transitivity, if we define that  $x$  is equal to  $y$ , then we have to connect to  $x$  all nodes incoming to  $y$  and using the same label (dotted lines on Fig. 4). Next, we take the “expanded” subword out of the word  $W$  and obtain the word  $V$ . In matrix notation the operation

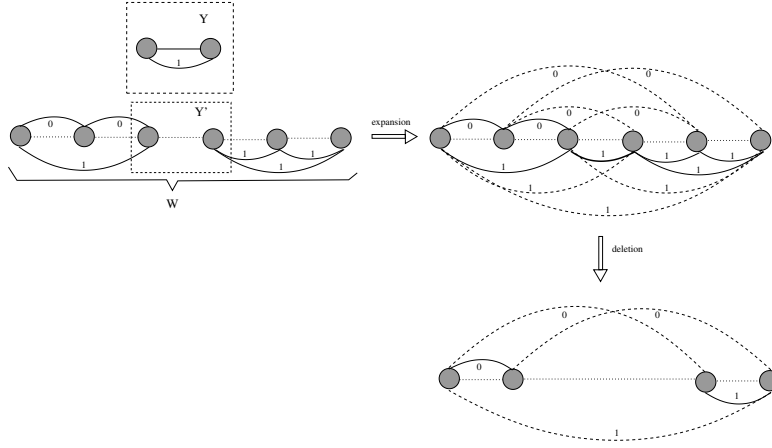


Figure 4: The single-step deletion relation  $W \xrightarrow[S]{\text{del}} V$

from Fig. 4 can be represented as:

$$\left( \begin{array}{cc|cc|cc} 1 & 0 & 1 & 2 & 2 & 2 \\ 0 & 1 & 0 & 2 & 2 & 2 \\ \hline 1 & 0 & 1 & 2 & 2 & 2 \\ 2 & 2 & 2 & 1 & 1 & 1 \\ \hline 2 & 2 & 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 1 & 1 & 1 \end{array} \right) \xrightarrow[Y]{\text{del}_3} \left( \begin{array}{cccc} 1 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \\ 0 & 2 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{array} \right), \text{ where } Y = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Formally the single-step deletion relation is defined as follows.

**Definition 8.** The single-step deletion relation on  $\mathbb{RW}$  that is induced by  $S = (INS, DEL)$  is defined as follows. For any  $W, V \in \mathbb{RW}$ ,  $Y \in Del$  and a integer  $1 \leq k \leq |W|$  we have  $W \xrightarrow[Y]{\text{del}_k} V$  if

- $(i, j) \in R^W$  implies  $(s_{k-1, m}^{-1}(i), s_{k-1, m}^{-1}(j)) \in R^V$ , where  $m = |Y|$  and  $R \in \{E, N\}$ ,
- $(i, j) \in R^Y$  implies
  - either  $(i + k - 1, j + k - 1) \in R^W$ , where  $1 < i, j \leq |Y|$ ,  $R \in \{E, N\}$ ,
  - or  $(i + k - 1, j + k - 1) \in U^W$  and for all pairs  $(i + k - 1, q) \cup (p, j + k - 1) \in E^W$  it holds  $(s_{k-1, m}^{-1}(p), s_{k-1, m}^{-1}(q)) \in R^V$  ( $R \in \{E, N\}$ ),
  - or  $(i + k - 1, j + k - 1) \in U^W$ ,  $R = E$  and for all pairs  $(i + k - 1, q) \in E^W$ ,  $(i + k - 1, q') \in N^W$ ,  $(p, j + k - 1) \in E^W$ ,  $(p', j + k - 1) \in N^W$  it holds  $(s_{k-1, m}^{-1}(p), s_{k-1, m}^{-1}(q)) \in N^V$  and  $(s_{k-1, m}^{-1}(p'), s_{k-1, m}^{-1}(q')) \in N^V$ .



As for insertion we will write  $W \xrightarrow[S]{\text{del}} V$ , meaning that there exists  $Y \in DEL$  and  $k \geq 1$  such that  $W \xrightarrow[Y]{\text{del}^k} V$ .

**Definition 9.** The deletion relation on  $\mathbb{RW}$  that is induced by  $S = (INS, DEL)$  is the reflexive, transitive closure of  $\xrightarrow[S]{\text{del}}$  and is denoted by  $\xrightarrow[S]{\text{del}^*}$ .

Each of the relations  $\xrightarrow[S]{\text{ins}}$  and  $\xrightarrow[S]{\text{del}}$  is denoted by  $\xRightarrow[S]$  and the reflexive, transitive closure of  $\xRightarrow[S]$  is denoted by  $\xRightarrow[S]^*$ .

**Definition 10.** An insertion-deletion system is the tuple  $S = (V, INS, DEL, A)$ ,  $V$  is an alphabet,  $(INS, DEL)$  is an insertion-deletion scheme and  $A \subseteq V^*$  is the initial language (the axioms) of the system.

If  $A = \emptyset$  then we will use a shorthand notation denoting the corresponding system as  $S = (INS, DEL)$ , i.e. we will identify it by the corresponding insertion-deletion scheme.

**Definition 11.** For an insertion-deletion system  $S = (V, INS, DEL, A)$  we define the language set  $L(S) = \{W \in \mathbb{RW} \mid Z \xRightarrow[S]^* W, Z \in A\}$  and the set  $FDL(S) = \{W \in \mathbb{FDRW} \mid Z \xRightarrow[S]^* W, Z \in A\}$ .

### 3 Main results

**Lemma 1.** For every insertion-deletion system  $S$  and every  $W, V \in \mathbb{RW}$  if  $W \xRightarrow[S]^* V$ , then there is  $Y \in \mathbb{RW}$  such that  $W \xrightarrow[S]{\text{ins}}^* Y \xrightarrow[S]{\text{del}}^* V$ .

*Proof.* First we prove that if  $W \xRightarrow[S]^* V$  and there are  $W_1, W_2, W_3 \in \mathbb{RW}$  such that  $W \xRightarrow[S]^* W_1 \xrightarrow[S]{\text{del}} W_2 \xrightarrow[S]{\text{ins}} W_3 \xRightarrow[S]^* V$  then there is  $W_2' \in \mathbb{RW}$  such that  $W \xRightarrow[S]^* W_1 \xrightarrow[S]{\text{ins}} W_2' \xrightarrow[S]{\text{del}} W_3 \xRightarrow[S]^* V$ , i.e., for any two consecutive operations of deletion and insertion in a derivation we can swap them so the insertion would be performed before the deletion.

Since  $W_1 \xrightarrow[S]{\text{del}} W_2$ , by definition of the operation  $\xrightarrow[S]{\text{del}}$  we have that there is a word  $Y \in DEL$  and a subword  $Y'$  in the word  $W_1$  such that  $Y'$  does not contradict  $Y$ . Let the word  $W_1$  has the length  $n$ , the subword  $Y'$  has the length  $m$  and starts from the position  $i$  in the word  $W_1$ . Then  $|W_2| = n - m$ . Since  $W_2 \xrightarrow[S]{\text{ins}} W_3$ , we have that there is a subword  $T$  in  $W_3$  and there is  $T' \in INS$  such that  $T = T'$ . Let  $T$  starts from the position  $j$  in  $W_3$ . We construct the word  $W_2'$  as follows. We take the word  $W_1$  and insert in it  $T'$  starting from the position  $j$  if  $j < i$  or from the position  $j + n$  if  $j \geq i$ . Then  $W_1 \xrightarrow[S]{\text{ins}} W_2'$ . Then the word  $W_2'$  has the scattered subword  $Y'$  that does not contradict  $Y$  since it was not changed during the insertion. Then we apply the deletion operation to  $W_2'$  deleting the subword  $Y$ . By the definition of the operation  $\xrightarrow[S]{\text{ins}}$  all the relations between symbols in  $W_1$  remain the same after insertion, and all symbols that have been inserted have undefined relationships with all other symbols in the

word. By the definition of the operation  $\xrightarrow[S]{\text{del}}$ , during the first step of deletion when we "expand" the subword  $Y$  to match  $Y'$ , we can change the relations between only those symbols which has relations with symbols from  $Y$  that are not undefined. Then all relations between symbols from subword  $T$  and all other symbols of the word after expansion remain unchanged, i.e. undefined, and all other relations are the same as the relations between corresponding symbols in  $W_3$ . Then we have that the result of deleting  $Y$  from  $W_2'$  is equal to  $W_3$ .

Thus for every derivation we can change  $W_1 \xrightarrow[S]{\text{del}} W_2 \xrightarrow[S]{\text{ins}} W_3$  to  $W_1 \xrightarrow[S]{\text{ins}} W_2' \xrightarrow[S]{\text{del}} W_3$ . By repeating this process for any place in the derivation where  $\xrightarrow[S]{\text{del}}$  goes directly before  $\xrightarrow[S]{\text{ins}}$ , we obtain the new derivation such that all  $\xrightarrow[S]{\text{ins}}$  precede all  $\xrightarrow[S]{\text{del}}$ , i.e.,  $W \xrightarrow[S]{\text{ins}}^* Y \xrightarrow[S]{\text{del}}^* V$ .  $\square$

Now we consider only simple insertion-deletion system from  $I_2D_3 \cup I_3D_2$ , i.e.,  $S = (INS, DEL)$  such that both sets  $INS$  and  $DEL$  contain only one rule and either the length of the insertion rule is 2 and the length of deletion rule is 3, or the length of the insertion rule is 3 and the length of deletion rule is 2.

Because of the transitivity of the relation  $E$ , there are only 2 different fully defined relational words of length 2 and 5 different fully defined relational words of length 3, yielding 10 insertion-deletion systems in both  $I_3D_2$  and  $I_2D_3$ . Below are the associated matrices.

$$M_1^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, M_2^2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, M_1^3 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, M_2^3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$M_3^3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, M_4^3 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, M_5^3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

**Lemma 2.** *For every simple insertion-deletion system  $S \in I_2D_3 \cup I_3D_2$  and every relational word  $W$  we have  $W \xrightarrow[S]{\text{ins}}^* \varepsilon$ .*

*Proof.* First we show that in each  $S \in I_3D_2$  we have  $V \xrightarrow[S]{\text{ins}}^* \varepsilon$  where  $|V| = 1$ .

Let  $S = (\{I\}, \{D\})$  where

1.  $I = \{M_1^3\}$ ,  $D = \{M_1^2\}$ . If we take the word  $V$ , insert the word  $I$  after the only symbol in  $V$ , and delete two last symbols in the result (they are equal), then we obtain the word of length two where the relation between symbols is undefined. Then we can delete these two symbols as equal and thus get the empty word. Hence we have following derivation:  $V \xrightarrow[I]{\text{ins}_1} V_1 \xrightarrow[D]{\text{del}_3} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$ . The derivations in matrix notation for this case and all the following cases can be found in the appendix.
2.  $I = M_1^3$ ,  $D = M_2^2$ . In this case we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[I]{\text{ins}_1} V_2 \xrightarrow[I]{\text{ins}_1} V_3 \xrightarrow[D]{\text{del}_7} V_4 \xrightarrow[D]{\text{del}_4} V_5 \xrightarrow[D]{\text{del}_3} V_6 \xrightarrow[D]{\text{del}_1} \varepsilon$ .
3.  $I = M_2^3$ ,  $D = M_1^2$ . In this case we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[I]{\text{ins}_2} V_2 \xrightarrow[I]{\text{ins}_2} V_3 \xrightarrow[D]{\text{del}_2} V_4 \xrightarrow[D]{\text{del}_5} V_5 \xrightarrow[D]{\text{del}_3} V_6 \xrightarrow[D]{\text{del}_1} \varepsilon$ .
4.  $I = M_2^3$ ,  $D = M_2^2$ . Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_1} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$ .
5.  $I = M_3^3$ ,  $D = M_1^2$ . Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_1} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$ .

6.  $I = M_3^3, D = M_2^2$ . Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_2} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$ .
7.  $I = M_4^3, D = M_1^2$ . Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_2} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$ .
8.  $I = M_4^3, D = M_2^2$ . Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_2} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$ .
9.  $I = M_5^3, D = M_1^2$ . Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[I]{\text{ins}_0} V_2 \xrightarrow[I]{\text{ins}_2} V_3 \xrightarrow[D]{\text{del}_6} V_4 \xrightarrow[D]{\text{del}_5} V_5 \xrightarrow[D]{\text{del}_2} V_6 \xrightarrow[D]{\text{del}_1} \varepsilon$ .
10.  $I = M_5^3, D = M_2^2$ . Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_1} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$ .

Thus in each  $S \in I_3D_2$  we have  $V \xRightarrow[S]{*} \varepsilon$ .

Since we can delete any isolated symbol, we can apply this sequence of rules to each symbol of the relational word  $W$  and thus we can delete the whole word, i.e., for each  $S \in I_3D_2$  we have that  $W \xRightarrow[S]{*} \varepsilon$ .

It is easy to see that in each  $S \in I_2D_3$  where  $S = (\{I\}, \{D\})$  the same sequence of rules that was used in  $S = (\{D\}, \{I\})$  to delete the isolated symbol but taken backwards adds to a relational word an isolated symbol. Then we can use the following strategy: we add to the end of the relational word  $W$  two isolated symbols and then apply to the last three symbols of the result the deletion rule, thus we obtain  $W$  shortened by one symbol. We repeat this process and delete all symbols in  $W$  one by one.

Thus we have that for every  $S \in I_2D_3 \cup I_3D_2$  and every relational word  $W$  we have  $W \xRightarrow[S]{*} \varepsilon$ .  $\square$

**Corollary 3.** *Let  $S$  be a simple insertion-deletion system such that  $S \in I_2D_3 \cup I_3D_2$ . For every fully defined relational words  $V$  and  $W$  we have  $V \xRightarrow[S]{*} W$  iff there is  $W' \in \mathbb{RW}$  such that  $W$  is a fully defined scattered subword of  $W'$  and  $V \xRightarrow[S]{*} W'$ .*

*Proof.* Let  $V \xRightarrow[S]{*} W$ . Since every relation word  $W$  has a fully defined subword  $W'$  of length 1 and by Lemma 2 for every  $W$  we can delete all the symbols of  $W$  but one, then  $W \xRightarrow[S]{*} W'$ .

Let  $W \in \mathbb{RW}$  be a fully defined scattered subword of  $W'$  and  $V \xRightarrow[S]{*} W'$ . Since by Lemma 2 we always can delete any symbol from the relation word without changing the relations between all other symbols, we have  $V \xRightarrow[S]{*} W' \xRightarrow[S]{*} W$ .  $\square$

In the next lemma we analyze the behavior of insertion-deletion systems in which all symbols in both insertion and deletion rules are equal. We prove that a relational word could be derived from the empty word if and only if all its symbols are equal.

**Lemma 4.** *Let  $S = (\{I\}, \{D\})$  be a simple insertion-deletion system such that  $S \in I_2D_3 \cup I_3D_2$  and for every  $x, y \in X^I$  we have  $(x, y) \in E^I$ , for every  $x', y' \in X^D$  we have  $(x', y') \in E^D$ , i.e., all symbols in both insertion and deletion rules are equal. For every relational word  $V$  we have  $V \in \text{FDL}(S)$  iff for every  $x, y \in X^V$  we have  $(x, y) \in E^V$ .*

To prove Lemma 4 we first show that by definitions of  $\xrightarrow[S]{\text{ins}}$  and  $\xrightarrow[S]{\text{del}}$  for every fully defined relational word  $V$  we have if  $V \in FDL(S)$ , then all the symbols in  $V$  are equal. Then we prove by induction on the length of the word that for every  $n \in \mathbb{N}$  we have  $\varepsilon \xrightarrow[S]^* V$  where  $V$  is a fully defined relational word of length  $n$  such that for every  $x, y \in X^V$  we have  $(x, y) \in E^V$ . Thus for every relational word  $V$  we have  $V \in FDL(S)$  iff for every  $x, y \in X^V$  we have  $(x, y) \in E^V$ . The full proof of the lemma can be found in the appendix.

In the next lemma we show that if  $S \in I_2D_3 \cup I_3D_2$  and either  $I$  or  $D$  contains unequal symbols then there is a constant  $k \in \mathbb{N}$  such that for every relational word  $V$  that could be obtained in  $S$  from the empty word, the length of the longest fully defined scattered subword of  $V$  is not larger than  $k$ .

**Lemma 5.** *Let  $S = (\{I\}, \{D\})$  be a simple insertion-deletion system such that  $S \in I_2D_3 \cup I_3D_2$  and there are  $x, y \in X^I$  such that  $(x, y) \in N^I$ , or there are  $x', y' \in X^D$  such that  $(x', y') \in N^D$ . Then there is  $k \in \mathbb{N}$  such that for every relational word  $V$  if  $V \in L(S)$ , then  $\max FD(V) \leq k$ .*

*Proof.* Let  $S = (\{I\}, \{D\})$  be a simple insertion-deletion system such that  $S \in I_2D_3 \cup I_3D_2$  and  $V$  be a relational word such that  $V \in L(S)$ , i.e.,  $\varepsilon \xrightarrow[S]^* V$ .

Then by Lemma 1 we have that there are  $V_0, V_1, V_2, \dots \in \mathbb{RW}$  such that  $\varepsilon \xrightarrow[S]{\text{ins}}^* V_0 \xrightarrow[S]{\text{del}} V_1 \xrightarrow[S]{\text{del}} V_2 \xrightarrow[S]{\text{del}} \dots \xrightarrow[S]{\text{del}} V$ .

By the definition of the insertion relation we have that for every relational words  $X$  and  $Y$  if  $X \xrightarrow[S]{\text{ins}}^* Y$ , then  $\max E(Y) = \max(\max E(X), \max E(I))$ ,  $\max FD(Y) = \max(\max FD(X), \max FD(I))$ . Then it follows that  $\max E(V_0) = \max E(I)$ ,  $\max FD(V_0) = \max FD(I) = |I|$ .

Now we consider the relational word  $D$ . Since either  $I$  or  $D$  contains unequal symbols, there are three cases for  $D$ : (1) there are no equal symbols in  $D$ ; (2) all symbols in  $D$  are equal; (3)  $D$  contains both equal and unequal symbols.

It can be shown by induction that

- in Case 1 there is a constant  $k = \max(|I|, |D| \cdot (\max E(I) - 1))$  such that for every  $V_i$ ,  $i \geq 1$  we have  $\max FD(V_i) \leq \max(|I|, |D| \cdot (\max E(I) - 1))$ . Since  $S \in I_2D_3 \cup I_3D_2$ , it is obvious that  $\max FD(V) \leq 4$ .
- in Case 2 for every  $V_i$  we have  $\max FD(V_i) \leq |I|$ . Since  $S \in I_2D_3 \cup I_3D_2$ , we have  $|I| \leq 3$ , then  $\max FD(V) \leq 3$ .
- in Case 3 for every  $V_i$  we have  $\max FD(V_i) \leq 3$ , then  $\max FD(V) \leq 3$ .

The full proof of the lemma can be found in the appendix.  $\square$

**Corollary 6.** *If  $S = (\{I\}, \{D\})$  be a simple insertion-deletion system such that  $S \in I_2D_3 \cup I_3D_2$  and there are  $x, y \in X^I$  such that  $(x, y) \in N^I$ , or there are  $x', y' \in X^D$  such that  $(x', y') \in N^D$ , then the set  $FDL(S)$  is finite.*

*Proof.* It follows from Lemma 5 that for every  $S \in I_2D_3 \cup I_3D_2$  the number of different fully defined scattered subwords in the set of all words that could be obtained from the empty word in  $S$  is finite since the length of such subwords is bounded by  $k$ . Then by Corollary 3 the set  $FDL(S)$  of all fully defined relational words that could be obtained in  $S$  is finite.  $\square$

**Theorem 7.** *Given a simple insertion-deletion system  $S \in I_2D_3 \cup I_3D_2$  and fully defined relational word  $V$ , it is decidable, whether  $V \in L(S)$ , i.e., whether  $\varepsilon \xRightarrow[S]{*} V$ .*

*Proof.* Let  $V$  be a fully defined relational word and  $S = (\{I\}, \{D\})$  be a simple insertion-deletion system such that  $S \in I_2D_3 \cup I_3D_2$ . Then there are 2 cases:

1. All symbols in both insertion and deletion rules are equal, i.e., for every  $x, y \in X^I$  we have  $(x, y) \in E^I$  and for every  $x', y' \in X^D$  we have  $(x', y') \in E^D$ ;
2. Either  $I$  or  $D$  contains unequal symbols, i.e., there are  $x, y \in X^I$  such that  $(x, y) \in N^I$ , or there are  $x', y' \in X^D$  such that  $(x', y') \in N^D$ .

In the first case by Lemma 4 we have that  $V \in L(S)$  iff for every  $x, y \in X^V$  we have  $(x, y) \in E^V$ . Then it is obvious that it is decidable, if  $V \in L(S)$ .

In the second case by Lemma 5 and Corollary 6 we have that the set  $L(S)$  is finite and there is a constant  $k$  that depends only on parameters of  $S$  such that each word in  $FDL(S)$  is not longer than  $k$ . Then we can obtain all the words in  $FDL(S)$  in finite time by building the derivation tree.  $\square$

## 4 Universality

In this section we show that if the length of the inserted and deleted words can be large, then corresponding insertion-deletion systems can produce a coding of any recursively enumerable language. We will abuse the terminology and we will call a function  $f : \mathcal{A}^* \rightarrow \mathbb{RW}$  (where  $\mathcal{A}$  is an alphabet) a morphism, if it satisfies  $f(uv) = f(u)f(v)$ . We will further restrict this notion and consider only those morphisms having  $f(a) \in \mathbb{FDRW}$ , for any  $a \in \mathcal{A}$ . Since any  $w \in \mathbb{FDRW}$  can be uniquely identified by a string, we will use such a representation to define corresponding morphisms. Notice, that  $f(u) \notin \mathbb{FDRW}$  for  $|u| > 1$ .

**Theorem 8.** *For any recursively enumerable language  $\mathcal{L}$  over a finite alphabet  $\mathcal{A}$  and for any (possibly infinite) alphabet  $\mathcal{V}$  with  $|\mathcal{V}| > 2$ , there exists an insertion-deletion system over relational words  $S = (\mathcal{V}, INS, DEL, A)$  and a morphism  $h$  such that  $\mathcal{L} = h^{-1}(L(S))$ .*

*Proof.* It is known that any recursively enumerable language can be generated by a context-free insertion-deletion system using strings over a finite alphabet [9]. This can be achieved by insertion of words of length 3 (resp. 2) and deletion of words of length 2 (resp. 3). Let  $S' = (V', T', INS', DEL', A')$  be such kind of system with  $L(S') = \mathcal{L}$ . We recall that  $L(S')$  contains words over  $T'$  reachable from the axioms of  $A'$ .

Let  $c : \mathcal{A} \rightarrow \mathbb{FDRW}$  be the morphism defined as follows:  $c(\mathfrak{a}_i) = (ab)^K a^i (ba)^K$ ,  $1 \leq i \leq n$ , where  $n = |\mathcal{A}|$  and  $K > n + 2$ . We will call  $c(\mathfrak{a}_i)$  the *code* of the letter  $\mathfrak{a}_i$ . We say that  $w \in \mathbb{RW}$  is in *canonical* form if  $c^{-1}(w) \neq \emptyset$ . Consider the extension of  $c$  to languages and let  $INS = c(INS')$ ,  $DEL = c(DEL')$  and  $A = c(A')$ . We also define  $h(a) = c(a)$ , if  $a \in T'$ .

We claim that  $\mathcal{L} = h^{-1}(L(S))$ . Clearly, due to the construction of  $S$  we immediately obtain that  $L(S)$  contains the image by  $c$  of all sentential forms used to obtain a word from  $L(S')$ . Next, we remark that the inverse morphism

$h^{-1}$  permits to select only relational words in canonical form corresponding to the concatenation of codes of terminal letters from  $T'$ , therefore its application yields a word from  $L(S')$ . Thus we obtain that  $\mathcal{L} \subseteq h^{-1}(L(S))$ .

In order to show the converse inclusion we need to prove that no other words except those corresponding to derivations of  $S'$  can be obtained. This can be formalized as follows.

**Claim 9.** *For any derivation  $w_1 \Rightarrow \dots \Rightarrow w_n$ ,  $w_i \neq w_j$ ,  $1 \leq i < j \leq n$  in  $S$ , if  $w_n$  is in canonical form, then any  $w_k$ ,  $1 \leq k < n$  is in canonical form.*

In order to prove the above assertion we will show that having  $w \Rightarrow w'$ , with  $w$  being in canonical form and  $w'$  not being in canonical form, implies that  $w' \not\Rightarrow^* w''$  with  $w''$  being in canonical form ( $w'' \neq w$ ). We shall prove this statement by contradiction. To simplify the arguments we assume that the sequence of derivations leading from  $w'$  to  $w''$  does not contain repeated words. We remark that  $w'$  can only be obtained by an insertion from  $w$  at a position not corresponding to the codeword boundary. Hence, in order to obtain a canonical word it is needed to “break” a sequence of codewords into pieces by insertion and to reconstruct new different codewords from these pieces. Since the deletion operation is performed for words in canonical form, a subword in canonical form should be obtained using the insertion operation.

We recall that each codeword  $c(\mathbf{a}_i)$  is composed from 3 different parts: the left part  $-(ab)^K$ , the middle part  $-a^i$  and the right part  $-(ba)^K$ . Since these parts can never match each other, the only way to obtain a subword in canonical form is to construct it symbol by symbol by a nested insertion of at most  $i + 2K$  codewords, each insertion should be done after the first (or before the last) symbol of each newly inserted word. We remark that this method permits to construct any sequence of symbols. However, since it takes the first letter away (the case of the last letter is similar), the remaining  $i + 2K$  left parts will contain the sequence  $b(ab)^{K-1}$ . Such a pattern can be completed to a codeword using the method above, but this introduces at least  $2K(K-1)$  same patterns  $b(ab)^{K-1}$ . Another possibility to complete it is to insert a codeword and to use its rightmost letter  $a$ , but this yields to the formation of the pattern  $(ba)^{K-1}b$ , hence the number of incorrect patterns does not change.

To conclude, in order to construct a pattern corresponding to a single codeword  $c(\mathbf{a}_i)$  at least  $i + 2K$  “incorrect” patterns (that need to be completed to a codeword) are generated. Moreover, the generation is performed in a nested manner, so no parts of it can form a subword in canonical form. Since each “completion” step introduces more words to be completed, this process can never lead to a word in a canonical form.

Now to conclude the proof of the theorem we remark that if every derivation in  $S$  is using words in canonical form, then this directly corresponds to a derivation in  $S'$ . Hence, no new words can be obtained yielding  $\mathcal{L} \supseteq h^{-1}(L(S))$ , which concludes the proof.  $\square$

Since the membership problem for recursively enumerable languages is undecidable we obtain the following corollary.

**Corollary 10.** *Given an insertion-deletion system  $S = (\mathcal{V}, INS, DEL, A)$  and a relational word  $X$ , it is undecidable, whether  $X \in L(S)$ , i.e., whether  $Z \xRightarrow[S]{*} X$ ,  $Z \in A \cup \{\varepsilon\}$ .*

**Further remarks:** We propose below two extensions of the model of insertion-deletion on relational words introduced in this paper. First we remark that a rewriting rule  $u \rightarrow v$  can be seen as the deletion of  $u$  and an insertion of  $v$  at the corresponding place. So, with small technical changes, the definitions 6 and 8 can be combined into a single definition for the rewriting operation. We remark that in the case of rewriting, the counterpart of Theorem 8 becomes trivial as the synchronization of the insertion and the deletion operation allows only rewriting of adjacent codewords.

Another extension is to consider the counterpart of the contextual variants of the insertion and deletion operation on strings [12]. In this case, the insertion or the deletion is performed in a specific context. The definition 6 can be adapted by first combining the left and right contexts into a single word, using a pattern-matching step like in definition 8 and then inserting the new word at the position given by contexts and keeping the relations between the context and the inserted word. For example, a rule  $(a, ab, b)$  would find an occurrence of two unequal symbols in the word and then would insert exactly between them two symbols equal to the symbol at left (resp. right) of the current position. The deletion operation can be defined similarly. In the case of contextual insertion and deletion the counterpart of Theorem 8 is also trivial, because it is possible to use the codes of entire symbols as left and right context. This means, that the operations can only be performed if the codewords are adjacent, i.e. in canonical form.

## References

- [1] Bojanczyk, M., David, C., Muscholl, A., Schwentick T., Segoufin, L. *Two-variable logic on data words.*: ACM Transactions on Computational Logic. Volume 12, Issue 4, (2011)
- [2] Ahmed Bouajjani, Cezara Dragoi, Yan Jurski, Mihaela Sighireanu: Rewriting Systems over Nested Data Words. MEMICS 2009
- [3] Ahmed Bouajjani, Peter Habermehl, Yan Jurski, Mihaela Sighireanu: Rewriting Systems with Data. FCT 2007: 1-22
- [4] Nadime Francis, Luc Segoufin, Cristina Sirangelo: Datalog Rewritings of Regular Path Queries using Views. ICDT 2014: 107-118
- [5] V. Halava, T. Harju, and T. Karki, Relational codes of words Theoretical Computer Science, vol. 389, no. 1-2, pp. 237-249, 2007.
- [6] Kaminski, M., Francez, N. *Finite-memory automata.*: Theor. Comput. Sci. 134(2), 329–363, (1994).
- [7] P. Leupold, Partial words for DNA coding, Lecture Notes in Comput. Sci. 3384 (2005) 224-234
- [8] Manuel, A., Ramanujam, R. *Automata over Infinite Alphabets.* In: D’Souza D., Shankar P. (eds.) Modern Applications of Automata Theory, pp. 529–554. World Scientific (2012)

- [9] Margenstern, M., Paun, G., Rogozhin, Y., Verlan, S. *Context-free insertion-deletion systems.*: Theor. Comput. Sci. 330(2), 339–348 (2005)
- [10] Neven, F., Schwentick, T., Vianu, V. *Finite state machines for strings over infinite alphabets.*: ACM Trans. Comput. Log. 5(3), 403–435, (2004).
- [11] S. Muthukrishnan, K. Palem, Non-standard stringology: Algorithms and complexity, in: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of computing 1994, ACM Press, New York, 1994, pp. 770779
- [12] G. Păun, G. Rozenberg, and A. Salomaa. *DNA Computing: New Computing Paradigms*. Springer, 1998.
- [13] Petre, I., Verlan, S. *Matrix insertion-deletion systems.*: Theor. Comput. Sci. 456, 80–88 (2012)
- [14] Rafiq Saleh: On the Length of Knot Transformations via Reidemeister Moves I and II. RP 2012: 121-136
- [15] Segoufin, L. *Automata and Logics for Words and Trees over an Infinite Alphabet*. In: Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29. LNCS, vol. 4207, pp. 41–57. Springer (2006)

## A Appendix

*Proof of Lemma 2.* We prove that in each  $S \in I_3D_2$  we have  $V \xRightarrow[S]{*} \varepsilon$  where  $|V| = 1$ .

Case 1. Let  $S = (\{I\}, \{D\})$  where  $I = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ ,  $D = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ . In

this case we have following derivation:  $V \xRightarrow[I]{\text{ins}_1} V_1 \xRightarrow[D]{\text{del}_3} V_2 \xRightarrow[D]{\text{del}_1} \varepsilon$ , and in matrix

notation we have  $\begin{pmatrix} 1 \end{pmatrix} \xRightarrow[I]{\text{ins}_1} \begin{pmatrix} 1 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 2 & 1 & 1 & 1 \end{pmatrix} \xRightarrow[D]{\text{del}_3} \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \xRightarrow[D]{\text{del}_1} \varepsilon$

Case 2. Let  $S = (\{I\}, \{D\})$  where  $I = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ ,  $D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ .

In this case we have  $V \xRightarrow[I]{\text{ins}_0} V_1 \xRightarrow[I]{\text{ins}_1} V_2 \xRightarrow[I]{\text{ins}_1} V_3 \xRightarrow[D]{\text{del}_7} V_4 \xRightarrow[D]{\text{del}_4} V_5 \xRightarrow[D]{\text{del}_3} V_6 \xRightarrow[D]{\text{del}_1} \varepsilon$

and in matrix notation we have  $\begin{pmatrix} 1 \end{pmatrix} \xRightarrow[I]{\text{ins}_0} \begin{pmatrix} 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 1 \end{pmatrix} \xRightarrow[I]{\text{ins}_1} \begin{pmatrix} 1 & 2 & 2 & 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 2 & 2 & 2 & 1 & 1 & 2 \\ 1 & 2 & 2 & 2 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 \end{pmatrix} \xRightarrow[I]{\text{ins}_1} \varepsilon$





Then in matrix notation  $(1) \xrightarrow[I]{\text{ins}_0} \left( \begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ \hline 2 & 2 & 2 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_1} \left( \begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_1} \varepsilon$

Case 5. Let  $I = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ ,  $D = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ .

Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_1} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$

and in matrix notation we have  $(1) \xrightarrow[I]{\text{ins}_0} \left( \begin{array}{ccc|c} 1 & 1 & 0 & 2 \\ 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ \hline 2 & 2 & 2 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_1} \left( \begin{array}{cc} 1 & 1 \\ 1 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_1} \varepsilon$

$\varepsilon$

Case 6. Let  $I = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ ,  $D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ . Then we have  $V \xrightarrow[I]{\text{ins}_0}$

$V_1 \xrightarrow[D]{\text{del}_2} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$  and in matrix notation we have  $(1) \xrightarrow[I]{\text{ins}_0} \left( \begin{array}{ccc|c} 1 & 1 & 0 & 2 \\ 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ \hline 2 & 2 & 2 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_2}$

$\left( \begin{array}{cc} 1 & 2 \\ 2 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_1} \varepsilon$

Case 7. Let  $I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ ,  $D = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ .

Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_2} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$  and in matrix notation we have

$(1) \xrightarrow[I]{\text{ins}_0} \left( \begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 \\ \hline 2 & 2 & 2 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_2} \left( \begin{array}{cc} 1 & 2 \\ 2 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_1} \varepsilon$

Case 8. Let  $I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ ,  $D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ .

Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[D]{\text{del}_3} V_2 \xrightarrow[D]{\text{del}_1} \varepsilon$  and in matrix notation we have

$(1) \xrightarrow[I]{\text{ins}_0} \left( \begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 \\ \hline 2 & 2 & 2 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_3} \left( \begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right) \xrightarrow[D]{\text{del}_1} \varepsilon$

Case 9. Let  $I = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ ,  $D = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ .

Then we have  $V \xrightarrow[I]{\text{ins}_0} V_1 \xrightarrow[I]{\text{ins}_0} V_2 \xrightarrow[I]{\text{ins}_2} V_3 \xrightarrow[D]{\text{del}_6} V_4 \xrightarrow[D]{\text{del}_5} V_5 \xrightarrow[D]{\text{del}_2} V_6 \xrightarrow[D]{\text{del}_1} \varepsilon$ . In ma-

trix notation we have  $(1) \xrightarrow[\text{I}]{\text{ins}_0} \left( \begin{array}{ccc|c} 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 2 \\ 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 1 \end{array} \right) \xrightarrow[\text{I}]{\text{ins}_0} \left( \begin{array}{ccc|ccc} 1 & 0 & 1 & 2 & 2 & 2 & 2 \\ 0 & 1 & 0 & 2 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 & 2 & 2 & 2 \\ \hline 2 & 2 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 0 & 1 & 0 & 2 \\ 2 & 2 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 1 \end{array} \right) \xrightarrow[\text{I}]{\text{ins}_2}$

$$\left( \begin{array}{ccc|ccc} 1 & 0 & 2 & 2 & 2 & 2 \\ 0 & 1 & 2 & 2 & 2 & 2 \\ 2 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 0 & 1 & 0 & 2 \\ 2 & 2 & 1 & 0 & 1 & 2 \\ \hline 1 & 0 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \end{array} \right) \xrightarrow[\text{D}]{\text{del}_6} \left( \begin{array}{cccccc} 1 & 0 & 2 & 2 & 2 & 1 \\ 0 & 1 & 2 & 2 & 2 & 0 \\ 2 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 0 & 1 & 0 & 2 \\ 2 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 1 & 0 & 1 & 2 \\ 1 & 0 & 2 & 2 & 2 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \end{array} \right) \xrightarrow[\text{D}]{\text{del}_5}$$

$$\left( \begin{array}{ccc|ccc} 1 & 0 & 2 & 2 & 2 & 2 \\ 0 & 1 & 2 & 2 & 2 & 2 \\ 2 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 0 & 1 & 0 & 2 \\ 2 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \end{array} \right) \xrightarrow[\text{D}]{\text{del}_2} \left( \begin{array}{ccc|cc} 1 & 0 & 2 & 2 & 2 \\ 0 & 1 & 2 & 2 & 2 \\ 2 & 2 & 1 & 0 & 2 \\ 2 & 2 & 0 & 1 & 2 \\ 2 & 2 & 1 & 0 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array} \right) \xrightarrow[\text{D}]{\text{del}_1} \left( \begin{array}{cc} 1 & 2 \\ 2 & 1 \end{array} \right) \xrightarrow[\text{S}]{\text{del}} \varepsilon$$

Case 10. Let  $I = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ ,  $D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ .

Then we have  $V \xrightarrow[\text{I}]{\text{ins}_0} V_1 \xrightarrow[\text{D}]{\text{del}_1} V_2 \xrightarrow[\text{D}]{\text{del}_1} \varepsilon$ .

Then in matrix notation we have  $(1) \xrightarrow[\text{I}]{\text{ins}_0} \left( \begin{array}{ccc|c} 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 2 \\ 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 1 \end{array} \right) \xrightarrow[\text{D}]{\text{del}_1} \left( \begin{array}{cc} 1 & 2 \\ 2 & 1 \end{array} \right) \xrightarrow[\text{D}]{\text{del}_1}$

$\varepsilon$

□

*Proof of Lemma 4.* First we show that for every fully defined relational word  $V$  we have if  $V \in FDL(S)$ , then all the symbols in  $V$  are equal.

By Lemma 1 for every  $V$  we have that  $\varepsilon \xRightarrow[\text{S}]{*} V$  iff there is  $V' \in \mathbb{RW}$  such that  $\varepsilon \xRightarrow[\text{S}]{\text{ins}}^* V' \xRightarrow[\text{S}]{\text{del}}^* V$ . Since in  $I$  all symbols are equal, by the definition of  $\xRightarrow[\text{S}]{\text{ins}}$  we have that if a relational word  $V'$  is obtained from  $\varepsilon$  by any number of insertions of  $I$ , then there are no unequal symbols in  $V'$ . Since all the symbols in  $D$  are also equal, by the definition of  $\xRightarrow[\text{S}]{\text{del}}$  while applying the deletion rule we cannot define that some symbols are unequal. Thus if  $\varepsilon \xRightarrow[\text{S}]{*} V$  and  $V$  is fully defined, then for every  $x, y \in X^V$  we have  $(x, y) \in E^V$ .

Now we prove by induction on the length of the word that for every  $n \in \mathbb{N}$  we have  $\varepsilon \xRightarrow[\text{S}]{*} V$  where  $V$  is a fully defined relational word of length  $n$  such that for every  $x, y \in X^V$  we have  $(x, y) \in E^V$ .

For the induction base, we suppose that  $n = 1$ . If  $S \in I_2D_3$ , i.e., the rule  $I$  consists of two equal symbols and the rule  $D$  consists of three equal symbols, then to derive  $V$  from  $\varepsilon$  we have to apply the insertion rule twice and then the deletion rule once. The illustration for this case is Fig. 6.

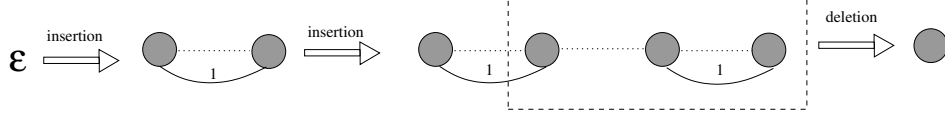


Figure 5: Obtaining  $V$  from  $\varepsilon$  when  $|V| = 1$  and  $S \in I_2D_3$

If  $S \in I_3D_2$ , i.e., the rule  $I$  consists of three equal symbols and the rule  $D$  consists of two equal symbols, then to derive  $V$  from  $\varepsilon$  we have to apply the insertion rule once and then the deletion rule once. The illustration for this case is Fig. 7.

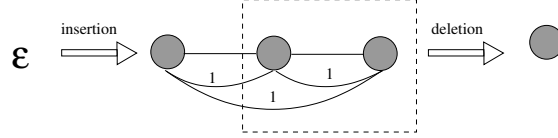


Figure 6: Obtaining  $V$  from  $\varepsilon$  when  $|V| = 1$  and  $S \in I_3D_2$

For the induction step, we assume that the hypothesis is true for all words of length  $n$  or less.

Let  $V$  be a word of length  $n$ . If  $S \in I_3D_2$ , then we make one insertion at the end of the word  $V$  and obtain the word of length  $n + 3$ . After that we delete the subword of two symbols - the last symbol of the word  $V$  and the first symbol of those which we have just inserted. The relation between these symbols is undefined, so we define them to be equal. Then by the definition of the deletion rule we define that all new symbols are equal to all the symbols of word  $V$ . After deleting two symbols we obtain a new word of length  $n + 1$  where all symbols are equal. This case is shown on Fig. 8.

If  $S \in I_2D_3$ , then we insert two equal symbols at the end of the word  $V$ , and then insert another two between them. After that we delete the subword of length three that starts from the last symbol of the word  $V$ , defining all undefined symbols to be equal. Then after deletion we obtain a new word of length  $n + 1$  where all symbols are equal. This case is shown on Fig. 9. This completes the induction step.  $\square$

*Proof of Lemma 5.* Let  $S = (\{I\}, \{D\})$  be a simple insertion-deletion system such that  $S \in I_2D_3 \cup I_3D_2$  and  $V$  be a relational word such that  $V \in L(S)$ , i.e.,  $\varepsilon \xRightarrow[S]{*} V$ . Then by Lemma 1 we have that there are  $V_0, V_1, V_2, \dots \in \mathbb{RW}$  such

that  $\varepsilon \xRightarrow[S]{\text{ins}}^* V_0 \xRightarrow[S]{\text{del}} V_1 \xRightarrow[S]{\text{del}} V_2 \xRightarrow[S]{\text{del}} \dots \xRightarrow[S]{\text{del}} V$ .

By the definition of the insertion relation we have that for every relational words  $X$  and  $Y$  if  $X \xRightarrow[S]{\text{ins}}^* Y$ , then

$$\max E(Y) = \max(\max E(X), \max E(I)) \quad (1)$$

$$\max FD(Y) = \max(\max FD(X), \max FD(I)) \quad (2)$$

Then we have

$$\max E(V_0) = \max E(I) \quad (3)$$

$$\max FD(V_0) = \max FD(I) = |I| \quad (4)$$

Now we consider the relational word  $D$ . Since either  $I$  or  $D$  contains unequal symbols, there are three cases for  $D$ :

1. There are no equal symbols in  $D$
2. All symbols in  $D$  are equal
3.  $D$  contains both equal and unequal symbols

**Case 1.** By the definition of the deletion relation, we have that if there are no equal symbols in  $D$ , i.e.,  $\max E(D) = 1$ , then during expansion step we never define that some symbols are equal.

Hence, for every relational words  $X$  and  $Y$  if  $X \xrightarrow[\text{S}]{\text{del}} Y$ , then  $\max E(Y) \leq \max E(X)$ .

Then for every  $V_i$  we have  $\max E(V_i) \leq \max E(V_0)$  and thus by (3)

$$\max E(V_i) \leq \max E(I) \quad (5)$$

Since there are no equal symbols in  $D$ , the longest possible fully defined subword of  $Y$  that we can obtain when we apply the deletion rule  $X \xrightarrow[\text{S}]{\text{del}} Y$  is of length  $|D| \cdot (\max E(X) - 1)$  (in this case there are  $|D|$  groups of equal symbols of size  $\max E(X)$  in the relational word  $X$ , and these groups are located in such a way that by applying one-step deletion we can define that the symbols of these groups are not equal). Thus if  $X \xrightarrow[\text{S}]{\text{del}} Y$ , then

$$\max FD(Y) \leq \max(\max FD(X), |D| \cdot (\max E(X) - 1)) \quad (6)$$

Then it could be easily shown by induction that for every  $V_i$ ,  $i \geq 1$  we have

$$\max FD(V_i) \leq \max(|I|, |D| \cdot (\max E(I) - 1)). \quad (7)$$

BASE STEP. Let  $i = 1$ , then by (6) we have

$$\max FD(V_1) \leq \max(\max FD(V_0), |D| \cdot (\max E(V_0) - 1)). \quad (8)$$

Since by (4)  $\max FD(V_0) = |I|$  and by (3)  $\max E(V_0) = \max E(I)$ , we have  $\max FD(V_1) \leq \max(|I|, |D| \cdot (\max E(I) - 1))$ .

INDUCTION STEP. Let us assume that there is  $i \geq 1$  such that for every  $j \leq i$  we have  $\max FD(V_j) \leq \max(|I|, |D| \cdot (\max E(I) - 1))$ . Then by (6) we have  $\max FD(V_{i+1}) \leq \max(\max FD(V_i), |D| \cdot (\max E(V_i) - 1))$ . Since by (5)  $\max E(V_i) \leq \max E(I)$ , we have  $\max FD(V_{i+1}) \leq \max(\max(|I|, |D| \cdot (\max E(I) - 1)), |D| \cdot (\max E(I) - 1))$ , i.e.  $\max FD(V_{i+1}) \leq \max(|I|, |D| \cdot (\max E(I) - 1))$ . This completes the induction step.

Thus for every  $V_i$  we have  $\max FD(V_i) \leq \max(|I|, |D| \cdot (\max E(I) - 1))$  and hence there is a constant  $k = \max(|I|, |D| \cdot (\max E(I) - 1))$  such that  $\max FD(V) \leq k$ . Since  $S \in I_2 D_3 \cup I_3 D_2$ , it is obvious that  $\max FD(V) \leq 4$ .

**Case 2.** In a similar way it can be shown that in the case when all the symbols in the deletion rule  $D$  are equal, for every  $i$  we have

$$\max E(V_{i+1}) \leq \max(\max E(V_i), |D| \cdot (\max E(V_i) - 1)), \quad (9)$$

$$\max FD(V_{i+1}) \leq \max(\max FD(V_i), \max FD(V_i) + (|D| - 1) \cdot \max E(V_i) - |D|). \quad (10)$$

We show by induction that for every  $V_i$  we have  $\max FD(V_i) \leq |I|$ .

BASE STEP. Let  $i = 1$ . By (9) we have  $\max E(V_1) \leq \max(\max E(V_0), |D| \cdot (\max E(V_0) - 1))$ . Since  $S \in I_2 D_3 \cup I_3 D_2$ , we have that either  $S \in I_2 D_3$  and  $|D| = 3$  or  $S \in I_3 D_2$  and  $|D| = 2$ .

If  $|D| = 2$  and all the symbols in the deletion rule  $D$  equal, then  $|I| = 3$  and  $\max E(I) \leq 2$ . Then  $\max E(V_1) \leq \max(\max E(I), 2) \leq 2$ .

If  $|D| = 3$  and all the symbols in the deletion rule  $D$  equal, then  $|I| = 2$  and  $\max E(I) \leq 1$ . Then  $\max E(V_1) \leq \max E(I) \leq 1$ .

By (10) we have  $\max FD(V_1) \leq \max(\max FD(V_0), \max FD(V_0) + (|D| - 1) \cdot \max E(V_0) - |D|)$ .

If  $|D| = 2$ , then  $\max E(V_0) \leq 2$  and  $\max FD(V_1) \leq \max FD(V_0) \leq |I|$ .

If  $|D| = 3$ , then  $\max E(V_0) \leq 1$  and again  $\max FD(V_1) \leq \max FD(V_0) \leq |I|$ .

INDUCTION STEP. Let us assume that there is  $i \geq 1$  such that for every  $j \leq i$  we have  $\max FD(V_j) \leq |I|$ .

Since by (9)  $\max E(V_{i+1}) \leq \max(\max E(V_i), |D| \cdot (\max E(V_i) - 1))$ , we have that if  $|D| = 2$ , then  $\max E(V_i) \leq 2$ , and if  $|D| = 3$ , then  $\max E(V_i) \leq 1$ .

Since by (10)  $\max FD(V_{i+1}) \leq \max(\max FD(V_i), \max FD(V_i) + (|D| - 1) \cdot \max E(V_0) - |D|)$ , we have that if  $|D| = 2$ , then  $\max FD(V_{i+1}) \leq |I|$ , and if  $|D| = 3$ , then again  $\max FD(V_{i+1}) \leq |I|$ .

Thus there is a constant  $k = |I|$  such that  $\max FD(V) \leq k$ . Since  $S \in I_2 D_3 \cup I_3 D_2$ , we have  $|I| \leq 3$ , then  $\max FD(V) \leq 3$ .

**Case 3.** Since  $S \in I_2 D_3 \cup I_3 D_2$  and  $D$  contains both equal and unequal symbols, then it is possible only when  $|D| = 3$  and  $D$  contain two equal symbols and the third symbol is not equal to them. In this case it can be shown that for every  $i$  we have

$$\max E(V_{i+1}) \leq \max(\max E(V_i), 2 \cdot (\max E(V_i) - 1)), \quad (11)$$

$$\max FD(V_{i+1}) \leq \max(\max FD(V_i), \max FD(V_i) + \max E(V_i) - 2, 3 \cdot (\max E(V_i) - 1)). \quad (12)$$

Since  $|D| = 3$ , it follows that  $|I| = 2$ , i.e.,  $\max E(V_0) \leq 2$  and  $\max FD(V_0) \leq 2$ .

We show by induction that for every  $i$  we have  $\max FD(V_i) \leq 3$ .

BASE STEP. Let  $i = 1$ . Then we have  $\max E(V_1) \leq \max(\max E(V_0), 2 \cdot (\max E(V_0) - 1))$ , i.e.,  $\max E(V_1) \leq 2$ .

Since  $\max FD(V_1) \leq \max(\max FD(V_0), \max FD(V_0) + \max E(V_0) - 2, 3 \cdot (\max E(V_0) - 1))$ , it follows that  $\max FD(V_1) \leq 3$ .

INDUCTION STEP. Let us assume that there is  $i \geq 1$  such that for every  $j \leq i$  we have  $\max FD(V_j) \leq 3$ .

Since for every  $i$  we have  $\max E(V_{i+1}) \leq \max(\max E(V_i), 2 \cdot (\max E(V_i) - 1))$  and  $\max E(V_0) \leq 2$ , it follows that for every  $i$  we have  $\max E(V_i) \leq 2$ .

Since  $\max FD(V_{i+1}) \leq \max(\max FD(V_i), \max FD(V_i) + \max E(V_i) - 2, 3 \cdot (\max E(V_i) - 1))$ , we have  $\max FD(V_{i+1}) \leq \max(3, 3 + \max E(V_i) - 2, 3 \cdot (\max E(V_i) - 1))$ . Then  $\max FD(V_{i+1}) \leq 3$ . Thus there is a constant  $k = 3$  such that  $\max FD(V) \leq k$ .

□

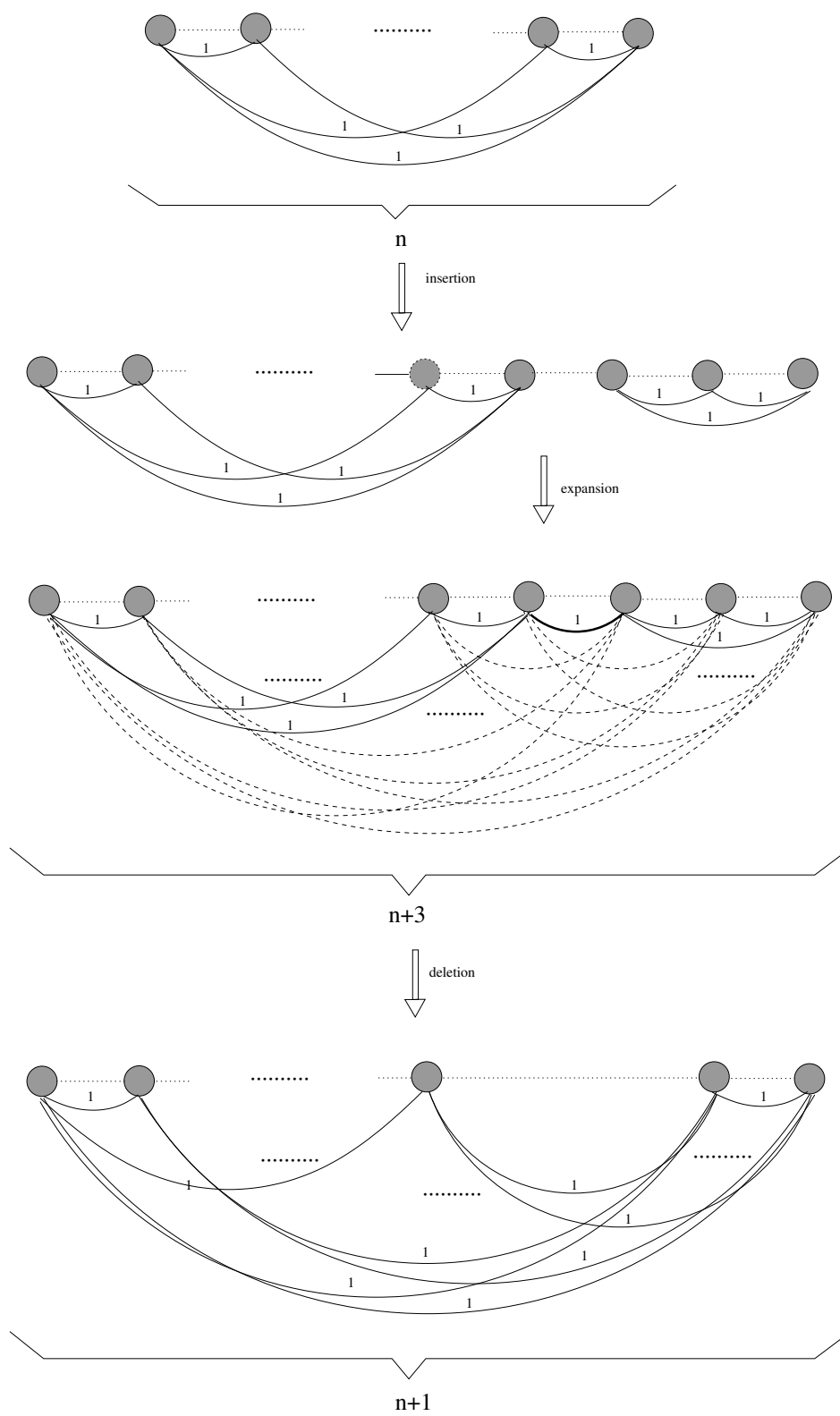


Figure 7:

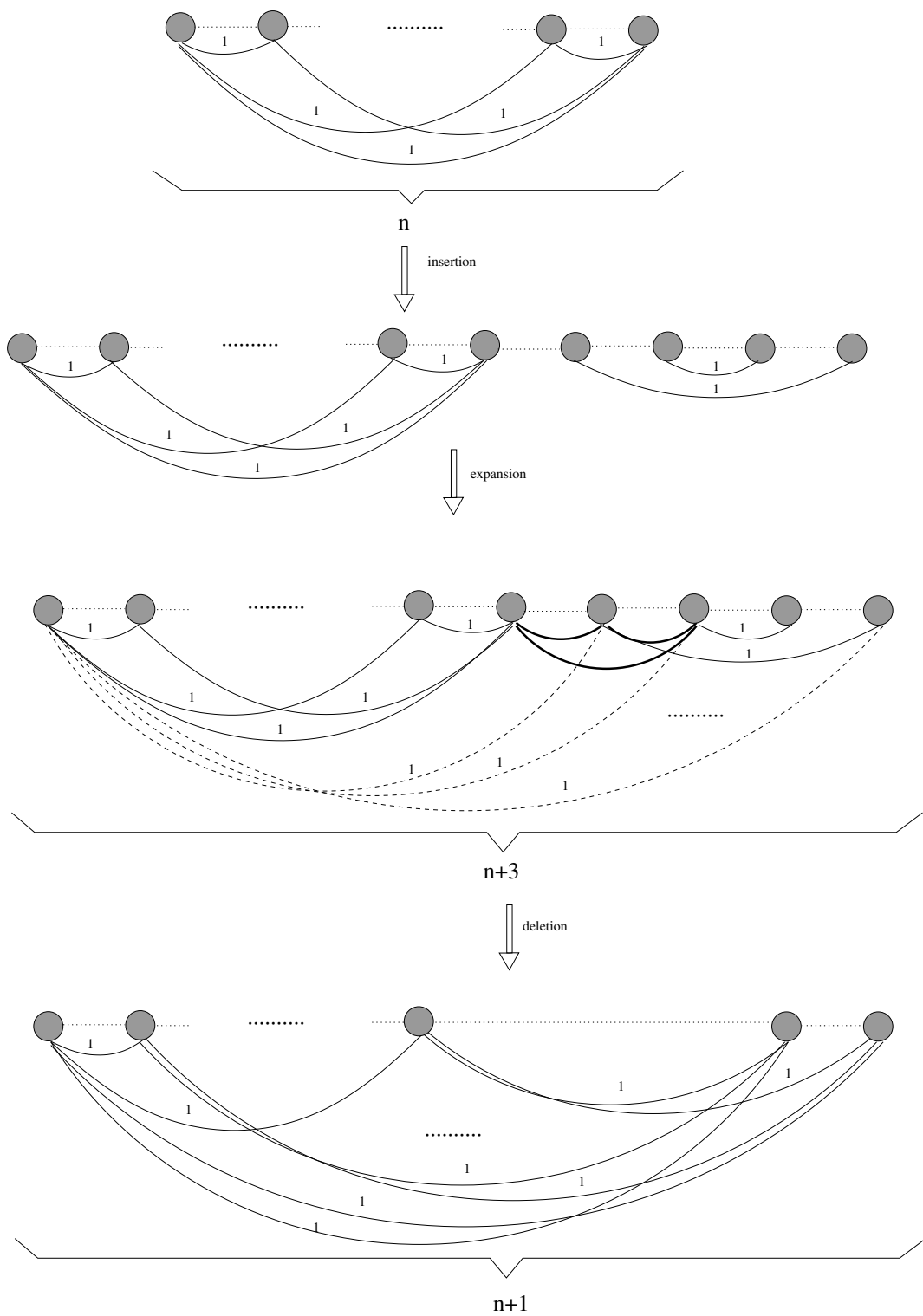


Figure 8: